Amendments to the Claims:

Re-write the claims as set forth below. This listing of claims will replace all prior versions and listings, of claims in the application:

Listing of Claims:

1. (currently amended) A unified shader comprising:

an input interface for receiving a packet from a rasterizer;

a shading processing mechanism configured to produce a resultant value from said packet by performing one or more shading operations, wherein said shading operations comprise both texture operations and color operations and comprising at least one ALU/memory pair operative to perform both texture operations and color operations wherein texture operations comprise at least one of: issuing a texture request to a texture unit and writing received texture values to the memory; and

an output interface configured to send said resultant value to a frame buffer.

2. (original) The shader of claim 1 wherein said input interface receives said packet from said rasterizer using a valid-ready protocol.

3. (original) The shader of claim 1 wherein said output interface sends said value to said frame buffer using a valid-ready protocol.

4. (original) The shader of claim 1 further comprising: a code partition mechanism to partition code configured to instruct said shading mechanism.

5. (original) The unified shader of claim 4 wherein said partitioning mechanism groups code together by level of indirection.

6. (previously presented) The unified shader of claim 5 further comprises control logic to process said partitioned code, wherein said control logic comprises:

an input state machine;

a plurality of ALU state machines; and

a plurality of texture machines.

7. (original) The unified shader of claim 1 further comprises: a register sub-system.

8. (original) The unified shader of claim 1 wherein said shading mechanism further comprises:

a plurality of ALU/memory pairs to perform said shading operations.

9. (original) The unified shader of claim 8 wherein said plurality of ALU/memory pairs

constitute a single coherent memory structure, wherein said plurality of ALU/memory are synchronized by a scheduling clock mechanism.

10. (original) The unified shader of claim 9 wherein said plurality of ALU/memory pairs constitute a pipeline for processing said shading operations.

11. (original) The unified shader of claim 9 wherein said wherein said memory structure is a FIFO that does not have an associated buffer.

12. (original) The unified shader of claim 11 wherein said FIFO comprises both data and operation instructions.

13. (canceled)

14. (previously presented) A method for shading comprising:

receiving a packet from a rasterizer;

obtaining a value by performing one or more shading operations using a plurality of ALU/memory pairs, wherein said shading operations comprise both texture operations and color operations and wherein each of the ALU/memory pairs performs both texture and color operations wherein texture operations comprise issuing a texture request to a texture unit and writing received texture values to the memory; and

sending said value to a frame buffer.

15. (original) The method of claim 14 wherein said receiving uses a valid-ready protocol.

16. (original) The method of claim 14 wherein said sending uses a valid-ready protocol.

17. (original) The method of claim 14 wherein said obtaining comprises: partitioning a code configured to instruct a unified shader.

18. (original) The method of claim 17 wherein said partitioning groups code together by level of indirection.

19. (previously presented) The method of claim 17 wherein said obtaining further comprises using control logic to process said partitioned code, wherein said control logic comprises:

an input state machine;

a plurality of ALU state machines; and

a plurality of texture machines.

20. (original) The method of claim 14 wherein said obtaining further comprises using a register sub-system.

21. (original) The method of claim 14 wherein said obtaining comprises: using a plurality of ALU/memory pairs to perform said shading operations.

22. (original) The method of claim 21 wherein said plurality of ALU/memory pairs constitute a single coherent memory structure, wherein said plurality of ALU/memory pairs are synchronized by a scheduling clock mechanism.

23. (original) The method of claim 22 wherein said plurality of ALU/memory pairs constitute a pipeline for processing said shading operations.

24. (original) The method of claim 22 wherein said memory structure is a FIFO that does not have an associated buffer.

25. (original) The method of claim 24 wherein said FIFO comprises both data and operation instructions.

26. (original) The method of claim 14 wherein said obtaining comprises: using a plurality of connected unified shaders, wherein said unified shaders are synchronized by a clock mechanism to process said shading operations together.

27. (currently amended) A computer program product comprising:

a physical computer usable storage medium~~, not including carrier waves,~~ having computer readable program code ~~embodied~~ stored therein configured to shade, said computer program product comprising:

computer readable code configured to cause a computer to receive a packet from a rasterizer;

computer readable code configured to cause a computer to obtain a value by performing one or more shading operations using a plurality of ALU/memory pairs, wherein said shading operations comprise both texture operations and color operations and wherein each of the ALU/memory pairs performs both texture and color operations

wherein texture operations comprise issuing a texture request to a texture unit and writing received texture values to the memory; and

computer readable code configured to cause a computer to send said value to a frame buffer.

28. (original) The computer program product of claim 27 wherein said computer readable code configured to cause a computer to receive uses a valid-ready protocol.

29. (original) The computer program product of claim 27 wherein said computer readable code configured to cause a computer to send uses a valid-ready protocol.

30. (original) The computer program product of claim 27 wherein said computer readable code configured to cause a computer to obtain comprises:

computer readable code configured to cause a computer to partition a code configured to instruct a unified shader.

31. (original) The computer program product of claim 30 wherein said computer readable code configured to cause a computer to partition groups code together by level of indirection.

32. (original) The computer program product of claim 31 wherein said computer readable code configured to cause a computer to obtain further comprises computer readable

code configured to cause a computer to use a control logic to process said partitioned code, wherein said control logic comprises:

an input state machine

a plurality of ALU state machines; and

a plurality of texture machines.

33. (original) The computer program product of claim 32 wherein said computer readable code configured to cause a computer to obtain further comprises computer readable code configured to cause a computer to use a register sub-system.

34. (original) The computer program product of claim 27 wherein said computer readable code configured to cause a computer to obtain comprises:

computer readable code configured to cause a computer to use a plurality of ALU/memory pairs to perform said shading operations.

35. (original) The computer program product of claim 34 wherein said plurality of ALU/memory pairs constitute a single coherent memory structure, wherein said plurality of ALU/memory pairs are synchronized by a scheduling clock mechanism.

36. (original) The computer program product of claim 35 wherein said plurality of ALU/memory pairs constitute a pipeline for processing said shading operations.

37. (original) The computer program product of claim 35 wherein said memory structure is a FIFO that does not have an associated buffer.

38. (original) The computer program product of claim 37 wherein said FIFO comprises both data and operation instructions.

39. (original) The computer program product of claim 27 wherein said computer readable code configured to cause a computer to obtain comprises:

compute readable code configured to cause a computer to use multiple unified shaders wherein said unified shaders are connected.

40. (previously presented) A device comprising:

a plurality of unified shaders synchronized by a clock mechanism to process shading operations together, wherein each of the unified shaders comprises:

an input interface for receiving a packet from a rasterizer;

a shading processing mechanism configured to produce a resultant value from said packet by performing one or more shading operations, wherein said shading operations comprise both texture operations and color operations and comprising at least one ALU/memory pair operative to perform both texture operations and color operations wherein texture operations comprise issuing a texture request to a texture unit and writing received texture values to the memory; and

an output interface configured to send said value to a frame buffer.

41. (previously presented) A unified shader comprising:

an input interface for receiving a packet from a rasterizer;

a shading processing mechanism configured to produce a resultant value from said packet by performing one or more shading operations, wherein said shading operations comprise both texture operations and color operations;

an output interface configured to send said value to a frame buffer;

a code partition mechanism to partition code configured to instruct said shading mechanism wherein said partitioning mechanism groups code together by level of indirection; and

control logic to process said partitioned code, wherein said control logic comprises:

an input state machine;

a plurality of ALU state machines; and

a plurality of texture machines.

42. (previously presented) A method for shading comprising:

receiving a packet from a rasterizer;

obtaining a value by performing one or more shading operations, wherein said shading operations comprise both texture operations and color operations;

sending said value to a frame buffer;

wherein said obtaining comprises: partitioning a code configured to instruct a unified shader;

wherein said obtaining further comprises uses a control logic to process said partitioned code and wherein said control logic comprises:

an input state machine;

a plurality of ALU state machines; and

a plurality of texture machines.

43. (previously presented) A computer program product comprising:

a computer usable medium, not including carrier waves, having computer readable program code embodied therein configured to shade, said computer program product comprising:

computer readable code configured to cause a computer to receive a packet from a rasterizer;

computer readable code configured to cause a computer to obtain a value by performing one or more shading operations, wherein said shading operations comprise both texture operations and color operations;

computer readable code configured to cause a computer to send said value to a frame buffer;

wherein said computer readable code configured to cause a computer to obtain comprises:

computer readable code configured to cause a computer to partition a code configured to instruct a unified shader;

wherein said computer readable code configured to cause a computer to partition groups code together by level of indirection;

wherein said computer readable code configured to cause a computer to obtain further comprises computer readable code configured to cause a computer to use a control logic to process said partitioned code, wherein said control logic comprises:

an input state machine

a plurality of ALU state machines; and

a plurality of texture machines.

44. (new) The unified shader of claim 1 wherein the memory of the at least one ALU/memory pair is controlled to store input values and intermediate variables needed by the shading processing mechanism and is allocated for a time it takes to process quad data and wherein the ALU of the ALU/memory pair writes a texture address to memory, writes a color value to the memory, reads a plurality of source operands from the memory and executes a shader instruction, writes a result from a previous shader instruction back to memory, reads a texture address from memory and issues it to a texture unit and writes a return texture value to the memory.